06 | 故障发现：如何建设On-Call机制？

2020-03-30 赵成

SRE实战手册

进入课程 >

讲述：赵成

时长 15:49　大小 10.88M

你好，我是赵成，从今天开始，我们进入课程实践篇的内容。

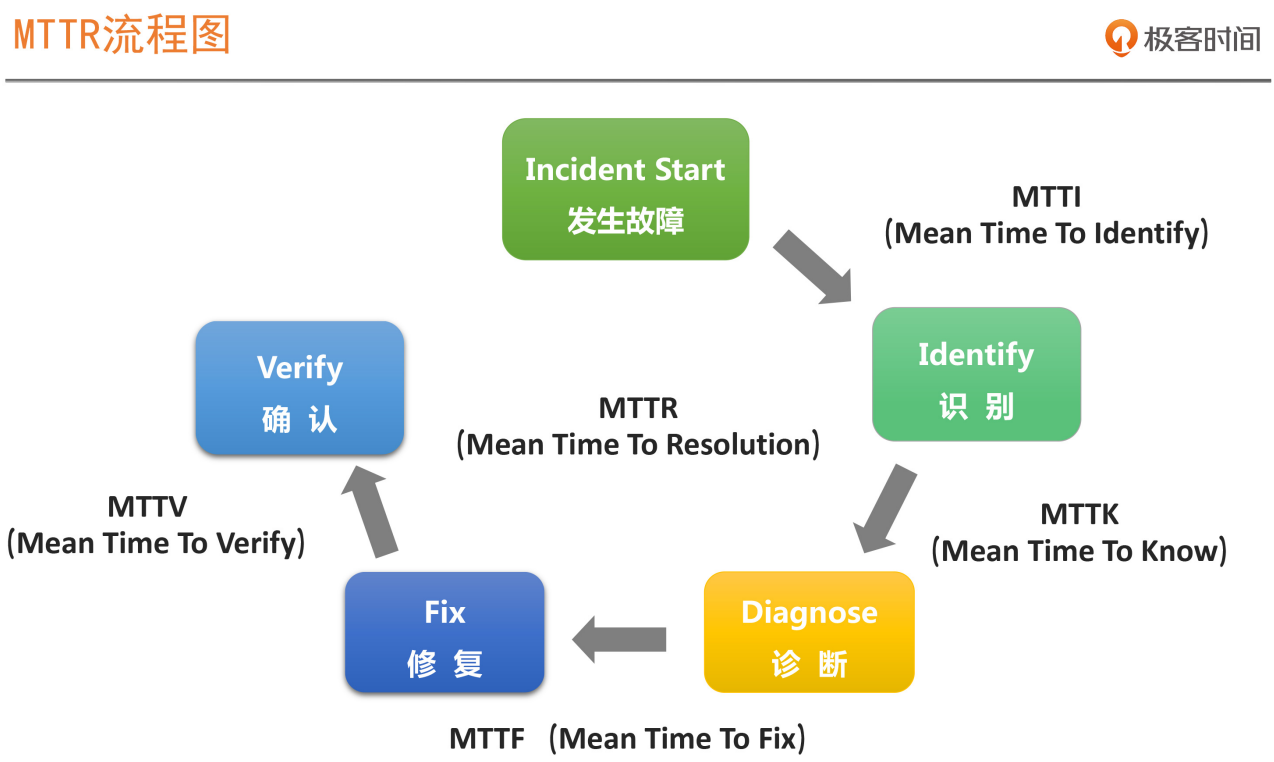
在上一部分，我们学习了 SRE 的基础，需要掌握的重点是 SLI 和 SLO 以及 Error Budget（错误预算）策略。SLI 是我们选择的衡量系统稳定性的指标，SLO 是每个指标对应的目标，而我们又经常把 SLO 转化为错误预算，因为错误预算的形式更加直观。转化后，我们要做的稳定性提升和保障工作，其实就是想办法不要把错误预算消耗完，或者不能把错误预算快速大量地消耗掉。

这么说，主要是针对两种情况：一种是我们制定的错误预算在周期还没有结束之前就消耗完了，这肯定就意味着稳定性目标达不成了；另一种情况是错误预算在单次问题中被消耗过多，这时候我们也要把这样的问题定性为故障。

今天，我们就来聊一聊，为了最大程度地避免错误预算被消耗，当我们定义一个问题为故障时，我们应该采取什么措施。

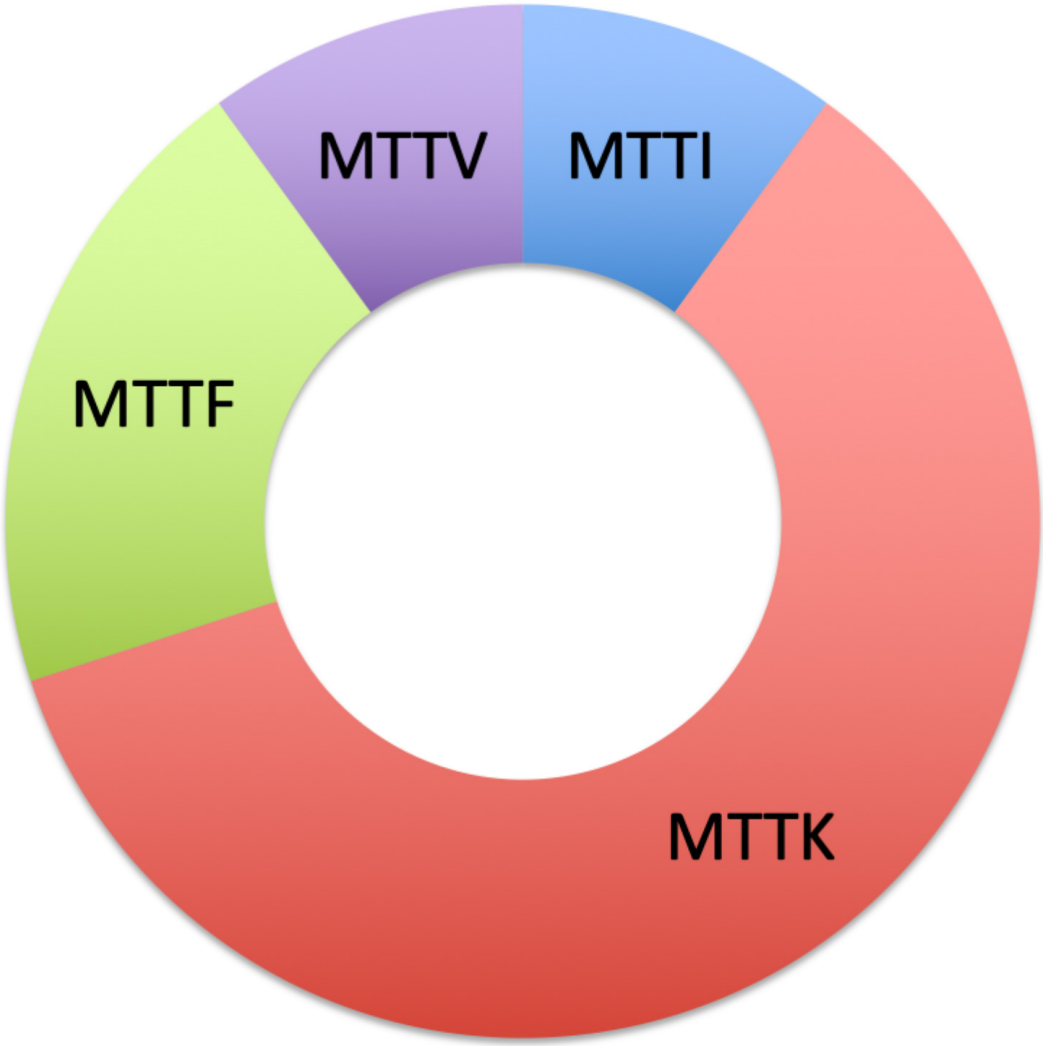
聚焦 MTTR，故障处理的关键环节

好了，我们先回顾下在第 1 讲的时候，提到故障处理的环节就是 MTTR，它又细分为 4 个指标：MTTI、MTTK、MTTF 和 MTTV，之所以分组分块，也是为了更加有目的性地做到系统稳定性。



那么，这四个环节，在我们故障处理 MTTR 又各自占多长时间呢？下面这个 MTTR 的时长分布图，是 IBM 在做了大量的统计分析之后给出的。

MTTR各环节所占时长（IBM）



我们可以看到，MTTK 部分，也就是故障定位部分的时长占比最大。这一点，我们应该都会有一些共鸣，就是绝大部分的故障，往往只要能定位出问题出在哪儿了，一般都可以快速地解决故障，慢就慢在不知道问题出在哪儿，所以说我们大部分时间都花在寻找问题上了。

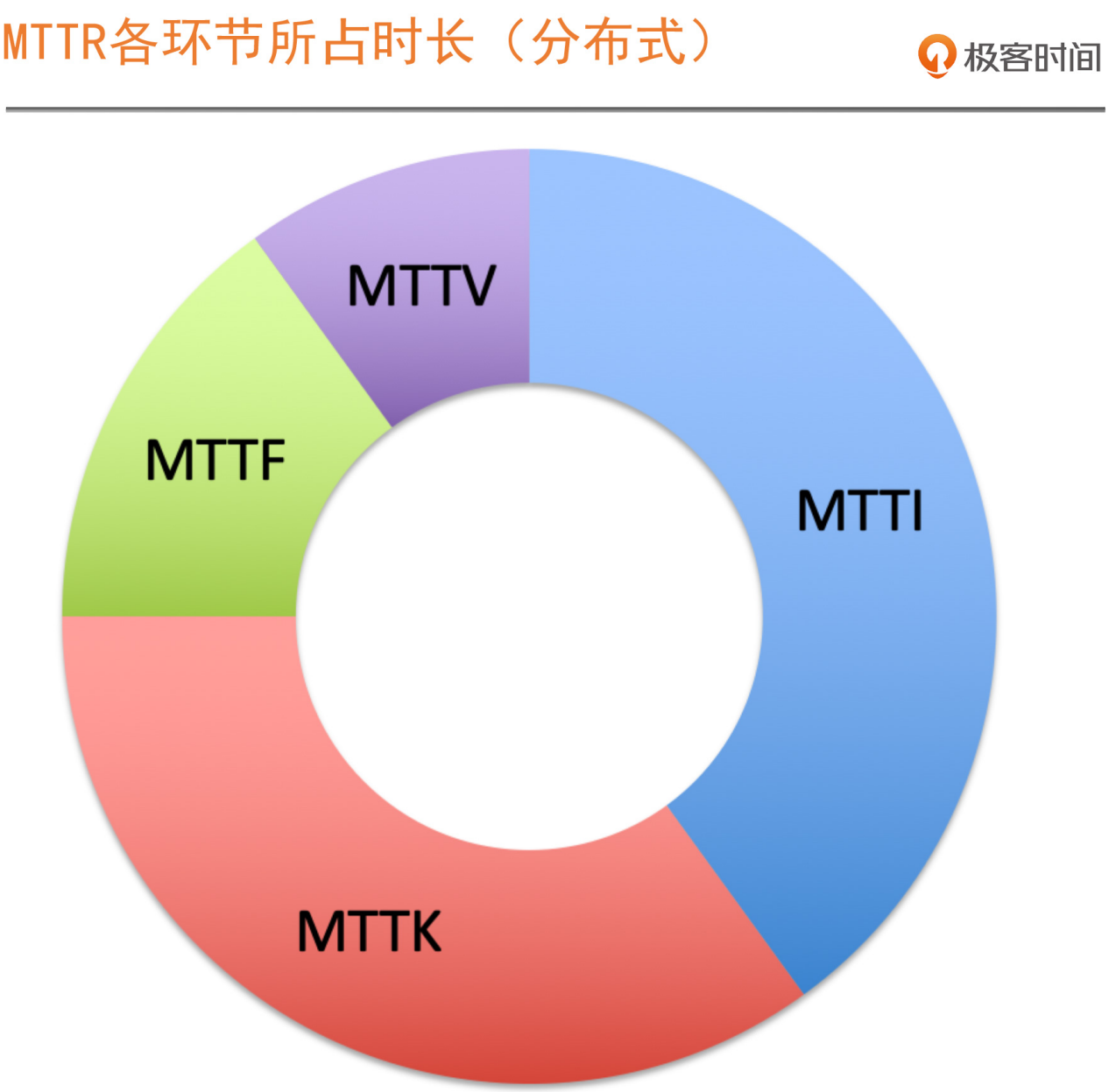
不过，从我实际分析的情况看，很多时候 MTTR 的时间分布跟这个图会有一些差别，为什么呢？

因为上面这张图是 IBM 针对网络设施的故障统计出来的，鉴于网络设备部署模式相对简单，以及日志和报错信息比较固定和明确，所以出现问题时，MTTI 的判定阶段不会花太多时间。而且真的出现问题，基本上采取的策略一般也是定位到根因，再采取措施处理故障，除了主备切换等冗余措施，基本是没有太多其它手段。

但是，在一个分布式的软件系统下，我们判定一个问题发生在哪儿，影响范围到底是怎么样的，要召集哪些人共同参与定位排查等等，这个反而会消耗更多时间，所以我们看到 MTTI 阶段占比会更重。

另外，当一个分布式系统发生故障时，我们的策略一定不是找到根因，而是优先恢复业务。所以，我们往往是通过故障隔离的方式，先快速恢复业务，也就是我们经常听到的 Design for Failure 这样的策略，再具体一点，就是服务限流、降级、熔断甚至是主备切换这样的手段，这样的话，MTTK 的占比会有所下降。

因此，从分布式系统的实际情况来看，整个 MTTR 的时间占比分布大致是这样的：



其实，不管是不是分布式系统，我们针对处理故障的目的都是比较明确的，就是要提升每个环节的处理效率，缩短它们的处理时间，这样才会缩短整个 MTTR，减少对不可用时长的消耗。

今天，我们就先来看 MTTR 的第一个环节，MTTI。看看在 MTTI 这个环节我们可以采取哪些措施来提高处理故障效率。关于 MTTR 中的另外三个环节，也就是 MTTK、MTTF 和 MTTV 我们会在后面的课程中继续讨论。

MTTI：从发现故障到响应故障

MTTI，就是故障的平均确认时间，也就是从故障实际发生时间点，到触发采取实际行动去定位前的这段时间。

这个环节其实主要有两件事情要做：**第一件事，判断出现的问题是不是故障；第二件事，确定由谁来响应和召集**。我们一一来看。

先看第一件事。我们平时遇到的问题很多，但是这些问题的影响不见得都很严重，或者都需要我们立即做出响应去处理。即便是故障，我们也会分不同的响应级别去处理。那我们怎么判断出现的问题是不是故障呢？如果是故障，我们应该以哪种级别去响应呢？

解决方案很明确，就是利用我们在《04 | 错误预算：达成稳定性目标的共识机制》中讲过的，根据错误预算制定故障等级的方式来判定响应方式。我们把 trade_cart 购物车的案例再拿出来，你可以再看下，作为参考。

故障等级设定

极客时间

Error Budget-错误预算	单次消耗比例	故障等级
25，000	比例<= 5%	P4
	5% < 比例 <= 20%	P3
	20% < 比例 <= 30%	P2
	30% < 比例 <= 50%	P1
	50% < 比例	P0

在没有 SLO 和错误预算这个体系时，我们通常会根据用户投诉，或客服对同一问题的重复反馈来判断。比如 10 分钟内，有超过 50 个用户投诉无法购买商品、支付不成功或者优惠券未生效等问题，我们就会启动应急响应。不过，一旦等用户和客服反馈，就说明故障影响已经非常恶劣了，用户也已经处于无法忍受的状态了。

显然这种方式并不高效。既然我们已经搭建了 SRE 体系，设定了 SLO，能对稳定性进行量化管理，那我们就能比用户更快发现问题并响应问题了。至于用户投诉和客服反馈的渠道，只能是作为我们的辅助手段。

所以，我们可以根据设定的 SLO 和错误预算，准确判断发生的问题是否是故障，并依据故障等级决定我们采取什么样的措施去处理这些问题，大大提高反应效率。

接着来看第二件事，谁来响应和召集。这件事很容易理解，如果我们判定这个问题就是一个故障，首先要有人能判定，接下来需要哪些人来介入，并能够把相关的人员召集起来，一起来处理故障。

这两件事，其实就是 SRE 里面提到的 **On-Call 机制**。

我们可以看到，第一件事其实主要依赖于我们的监控和告警体系。这里我想再强调一下，在 On-Call 阶段，并不是所有的告警我们都要响应，如果不影响稳定性，一般的告警是可以降

低响应级别的，我们只需要响应基于 SLO 的告警。

我和很多公司交流过，发现大家都非常重视监控体系的建设，监控指标梳理得也非常全面，而且各种监控技术和产品都会使用。比如，从最早的 Zabbix，以及日志系统栈 ELK，再到近两年随着 Kubernetes 而火爆起来的 Prometheus 等等，从技术上来说，这一点不是太大的问题。但是，第二件事往往容易被我们忽视，也就是 On-Call 的流程机制建设。

关于 On-Call 的两个案例

在正式讲流程机制之前，我先分享两个关于 On-Call 的案例。

第一个案例是发生在我自己团队里的，当时遇到一次大数据产品 HBase 故障。那是一个周六的下午，负责数据平台的运维同学收到了故障告警，出现了部分节点不可用，同时依赖该产品的搜索和推荐等业务也开始反馈异常，进而影响到广告曝光，导致资金损失。

问题出现后，我们很快明确了故障的严重程度，开始联系对应的负责人上线一起处理。因为是周末，所以需要一个个打电话联系，同时我们的 HBase 已经上云，所以处理过程又需要云产品的同事介入，这就又涉及到跨组织的协调。云产品内部又有分工，一线服务、二线产品研发支持，所以这个协调过程又是一个比较长的链条。再加上其他的权限问题，所以等相关人员就位开始真正排查问题的时候，距离发生故障已经有很长一段时间了。

当时我大致统计了下，仅仅协调人员到位并上线开始处理问题，这个过程就花去了 45 分钟，而真正解决问题的环节只用了 15 分钟。

第二个例子是国内第一大企业 IM 产品。因为产品本身的特点，每天早上八点半到九点，会有一波使用高峰，产品经常在这个时间段出故障。同时呢，这个时间段正好是互联网公司通勤时间，大部分技术员工都在上班的路上，根本没办法处理问题，导致故障时间过长。

所以为了保证及时响应，企业决定分批上下班，一批人早上在家值守，甚至是守在电脑旁随时应急，其他员工准时上班。等过了产品使用高峰期，值守的员工再陆续从家里出发上班。同时，下班也错开时间，保证有人可以及时响应。

这种错时保障机制，很多电商企业在大促保障或者重大活动期间也经常用，就是**确保关键角色必须在线，随时应急响应**。

通过以上两个案例，我们发现从整体上讲，如果要缩短 MTTI 时长，技术相关的监控告警很重要，但更关键的是一整套协作机制。这也是为什么像 Google 这样技术体系已经如此完善的公司，还要强调 On-Call 机制的重要性。

那么，我们怎样才能建设好 On-Call 的流程机制呢？

On-Call 的流程机制建设

接下来，我就跟你说说，在蘑菇街我们是怎么做的，我把这个流程总结为“**On-Call 关键 5 步法**”。

1.**确保关键角色在线。**

这里的关键角色不是指一两个值班的运维或 SRE 人员，而是整个产品体系中的所有关键角色。比如电商就需要安排核心业务应用（如用户、商品、交易、优惠及支付等）的 Owner，或 Backup 中至少有一人参与 On-Call 轮班。不过，接收故障告警或第一时间响应故障的，一般是运维、PE 或 SRE 这样的角色，业务开发同事要确保及时响应 SRE 发起的故障应急。

2.**组织 War Room 应急组织。**

我们有专门处理故障的“消防群”（暗含着灭火的意思），会将这些关键角色纳入其中，当有故障发生时会第一时间在消防群通报，这时对应的 On-Call 同事就要第一时间最高优先级响应呼叫（Page）。如果是工作日，对于严重故障，会立刻组织成立 War Room，责任人会马上聚到一起；如果是非工作时间，则会通过电话会议的方式拉起虚拟 War Room。

3.**建立合理的呼叫方式。**

在 On-Call 的落地过程中，我们经常会遇到的一种情况，就是谁最熟悉某个系统，谁就最容易被 7*24 小时打扰。比如系统或应用的 Owner 或者是架构师，出现问题的时候一定是找这些同事处理的效率最高，所以就会造成这些同事被默认为 On-Call。

但是这样做会极大地影响这些同事在正常业务开发上的精力投入。他们要么总是被打断，要么就是经常通宵处理问题，导致第二天无法正常工作，甚至在非工作日也得不到正常的休息。

这种情况下，我们建议使用固定的 On-Call 手机，建立手机与所有 On-Call 系统的对应关系，比如这个手机号码对应交易核心应用，另一个号码对应 IDC 机房运维等等。这样我们在 On-Call 时就不再找具体的哪个人，而是手机在谁手中，谁就承担 On-Call 职责。除非问题迟迟得不到解决，或者遇到了疑难杂症，这种时候再呼叫其他同事参与进来。

无论是 SRE、架构师，还是一线开发，**熟悉某个系统的最快最好的方式就是参与 On-Call，而不是看架构图和代码。**所以，有效的 On-Call 机制，可以让团队更深刻地认识到目前系统的存在哪些问题，对自己的痛苦状态也会有更深刻的感受，进而对后面的改进措施才能更有针对性和落地性。同时，On-Call 也可以培养和锻炼新人和 Backup 角色，这也是让新人尽快融入团队和承担责任的最好的方式。

4.**确保资源投入的升级机制。**

这个跟前面几条有相关性，有很多团队认为 On-Call 就是设置几个人值班，所有的事情都交给这几个人做；最极端的还会认为所有的事情，都应该是冲在最前线的运维或 SRE 来完成。但是，在分布式架构这种复杂场景下，这种思路是明显不可行的。

这里最核心的一点就是要给到运维和 SRE 授权，当他发现问题不是自己或现有 On-Call 人员能解决的时候，他有权调动其他必要资源投入。如果故障等级偏高，一下无法明确具体找哪些人员投入的时候，SRE 或运维可以直接上升到自己的主管或相关主管那里，要求上级主管协调资源投入。必要时，还可以上升到更高级主管，甚至 CTO 或技术 VP 来关注。所以，授权非常关键，这一点的具体操作层面，我们会在后面的故障处理过程中再详细介绍。

5.**与云厂商联合的 On-Call。**

现在企业上云是大势所趋，绝大多数情况下，我们对问题和故障的处理，是离不开与云产品工程师一起高效联动的。所以，我们应该把云产品和云厂商作为我们系统的一部分，而不是单纯的第三方。而对于云厂商来说，也要考虑把客户业务作为自身业务的一部分，只有这样双方才能紧密合作。我们应该提前做好与云厂商之间的协作磨合，联合故障演练，必要的授权等等。

总结

最后，我们做一下简单的总结。

我们按照 MTTR 的细化分段可以看到，处理故障的第一个环节就是 MTTI，也就是故障发现阶段。这个阶段，我们要靠基于 SLO 的告警，更加精准地告知我们当前系统的稳定性出现的问题，并根据对 SLO 的影响程度，也就是错误预算的消耗情况作出判断，是否定性成故障。如果是故障，我们就要启动应急响应，而高效快速的应急响应，是要靠 On-Call 的流程机制来保证的。

关于建设 On-Call 的流程机制，我给你分享了我自己团队的“On-Call 关键 5 步法”，咱们再一起复习一下：

1. 确保关键角色在线；
2. 组织 War Room 应急组织；
3. 建立合理的呼叫方式；
4. 确保资源投入的升级机制；
5. 与云的联合 On-Call。

当我们能够很好地做到以上两点的时候，我们就能大幅降低 MTTI，也为接下来更高效的故障定位和恢复打下了坚实的基础。

思考题

最后，给你留一个思考题。

学完本节课的内容后，你觉得在故障处理中，是优先建设监控体系更重要，还是建设高效的 On-Call 机制更重要？它们两者应该怎么来结合会更有效？欢迎你在留言区分享你的思考。

如果你在 On-Call 方面也积累了一些经验，请分享在留言区，也欢迎你把今天的内容分享给身边的朋友。

我是赵成，我们下节课见。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　05 | 案例：落地SLO时还需要考虑哪些因素?

下一篇　07 | 故障处理：一切以恢复业务为最高优先级

精选留言 (7)

写留言

美美 置顶

2020-03-31

谷歌SRE应急时间处理策略有一条是：由万（全）能工程师解决生产问题 向 手持“运维手册”的经过演练的on-call工程师 演进，核心思路是建设故障处理SOP，保证SRE可以处理大多数故障。这个思路是否和确保关键角色在线冲突!

展开

作者回复: “由万（全）能工程师解决生产问题 向 手持“运维手册”的经过演练的on-call工程师 演进，核心思路是建设故障处理SOP，保证SRE可以处理大多数故障”

感谢你的分享，直接道出了我们oncall应该追求的目标，对于我们内容是非常好的补充。

内容里我们提到的关键角色在线，就是指on-call工程师必须在值守过程中准实时在线，目的是为了及时响应，这个是机制上的保障，然后在处理问题时，需要有故障处理SOP，并且能按照SOP执行，这个是能力上的要求。

所以，两者不矛盾。再次感谢你的补充，非常棒。

1

moqi
2020-03-30

在某某公司的运维小伙和我提起过，他们那的系统出了问题，SRE的第一件事是写故障报告，第二件事是解决问题，坑爹的是解决问题的时候还得不停的回复boss们的询问，更坑爹的是其他的SRE没有任何的协作机制，看他一人在那忙死，没有丝毫的互备机制，老大们似乎也不觉着这是个巨大的问题。
...
展开

作者回复: 很好的关于感受的分享。

特别是第一段，应该是我们要极力避免的状态和情况。

2

一步
2020-03-30

关于On Call流程机制的建设说到了痛点，往往就是系统出问题了，不能迅速的联系对应的责任人

作者回复: 这个是通病，也是非常容易被忽视掉的部分，而忽视的原因，是因为跟技术没有太大关系。

1

leslie
2020-03-30

老师提及了错峰上班，可能我所经历的某些企业就是直接的翻班，这是国内IDC机房普遍使用的策略。必要时候这种翻班是从部门经理层一直到一线员工，集体翻班或者延长工作时间去保障其稳定性。
监控的体系再怎么做其实都是后知后觉，就像SRE中说及"没有故障是特殊现象 “，有问题有故障才是常态，故而On-call的机制是第一时间处理问题；二者又是相互的。只有在解...
展开

作者回复: 很好的分享。

因为IDC机房是基础设施，太关键了，很多银行和金融企业也是24小时现场值班，因为经不起半点疏忽。

监控和on-call在一段时间内是要互补的，监控跟不上，on-call机制来补，随着业务体量的增大，监控的重要性会越来越大，也要同步建设起来。

1

Christopher
2020-03-30

第一次听赵成老师的声音，还挺有磁性的，哈哈

展开

作者回复: 谢谢！

其实内容也很有料，是吗？^_^

1

Edward
2020-03-30

ON-call机制重要，毕竟监控体系是持续优化进步的过程，而业务的故障是随时发生，优先恢复业务是最高优先级。良好的on-call机制既能保障业务，也能将处理过的故障快速反馈到监控体系，进一步优化监控体系，达到双赢。

展开

作者回复: 思考的很全面奥

1

wholly
2020-03-30

看完老师的课程，很想从开发转岗到SRE，感觉很刺激😂

作者回复: 不经历磨难，怎么见光明。不一定要转岗，但是开发过程中要多跟SRE交流学习，时间长了，你就是具备SRE意识和能力的开发。

1